Email text to UI form

Ankur Sarda, Roy Shilkrot
Department of Computer Science
Stony Brook University
asarda@cs.stonybrook.edu, roys@cs.stonybrook.edu

Abstract

In this paper, we propose a solution to simplify email conversations. We break down an email to focus only on information being requested. Further we use these specific sentences and map it to HTML form elements, which can be quickly action-ed upon. We built a chrome extension to test the entire end-to-end experience on Gmail web-client.

1. Introduction

Most of the emails we reply to, require us to scan and analyze it and then provide a response. This becomes more cumbersome when the email text is long as we need to go back and forth to form an entire response. Sometimes, we forget to include an information or attachment that was requested. Due to enormous emails being sent everyday, we spend a lot of time performing this task. Also the chances of making a mistake is relatively high given it is created manually. In this paper, we do not intend to automate the task of providing a response to an email. Instead, provide a solution to make it very simple for the user to focus only on elements that require attention. And, augment it with a User Interface which can be used to provide quick replies.

An email can contain a variety of content. This content tends to be either informative or demands certain information which can be provided in the form of text or attachment. Also, the content of the email could be plain (text) or rich (HTML). Many a times an email has text as part of an inline image, making it hard to retrieve information. In this paper, we focus on emails only with mime type as text/plain.

The first task is to cleanse the email by removing signatures, punctuation and extract the main text body. We then break down the text into sentences using nltk[3]. Each sentence is then checked if it requests information from the user, using a series of classifiers (see 3.1 Question Analysis). The classifiers are built using scikit learn library[2]. The sentences marked positively are mapped to a particular UI element (see 3.2 UI Mapping). This is exposed as a rest service, which takes an email text as input and returns a list of tuples [Sentence, UI element, Fillers]. Fillers are extra information required to build UI.

A chrome extension tuned to work with Gmail web-client helps the user to provide only relevant information that is requested. A button, labeled as "Assistant" is provided to user which when clicked - extracts email content, invokes the rest service and creates a UI in a pop-up which contains an HTML form with textbox, radio buttons and other form elements (See 3.3 Chrome Extension).

2. Previous Work

Though there is no noticeable work in the field of mapping text to UI, but there has been numerous studies in question detection and its type identification. It has been studied primarily in the context of Question-Answering domain and email prioritization. In the field of question type detection we see one of the approaches taken is using multiple classifiers to refine the type of question[1]. Question-answering is used in different scenarios like question-answer pair detection in email conversations[5, 8], twitter conversations[6, 7]. For example, Shrestha and McKeown[5] learn a classification model based on parts-of-speech (POS) features, training it on a transcribed telephone speech corpus. Whereas Helen and Neil [8], use a combination of regex and machine learning for learning a model to detect imperative and declarative questions. In the field of prioritization, Mark Dredze, Tova Brook et. al. [1] studied reply and attachment prediction to relieve the stress of email overload. For reply prediction, they used a rule based predictor with relational and content specific features (bag of words). In this paper, we focus on bag of words with ngram size of 1 and 2 on a simple Naive Bayes multi-class classifier and create a UI from the email text using chrome extension.

3. Implementation

In this section we take a close look at the system and its various components.

3.1 Question Analysis

Email text is first segmented into sentences using "PunktSentenceTokenizer" of "nltk.tokenize.punkt module" in nltk NLP library. A model trained for English language sentences is used to perform the segmentation. All the punctuation symbols except question mark is removed.

Each sentence is then tested against three classifiers (A, B & C) to perform three different tasks.

Classifier A determines if a sentence is indeed a question. A question in any domain has a certain semantic. We do not require an expert to determine if a sentence is a question in a particular domain. But we cannot rely only on question mark as it is not an adequate measure. Additionally, a question can be posed in an informal language. Interrogative questions are more formal in nature. For example, Would you be free today at 5 PM?. The same sentence in a declarative form can be posed as "I was wondering if you are free today at 5 PM". Notice there is no question mark used in the latter.

To tackle this problem we use Machine Learning techniques to train a classifier using Amazon QA data set[4]. Other data sets that were examined were yahoo QA and Switch Board corpus. The choice of using Amazon QA data is due to its close resemblance to the email lingo where people tend to be both formal and informal. It also contains a mix of interrogative and declarative form of questions. People ask questions to other people on the forum about a product and it is aptly replied by other people in the community. This data set consists of question, type of question (Yes/No or open ended question) and an answer. A data set of 1500 questions and 1500 answers is picked at random with no repetition from this huge corpus. Further we choose sentences which has more than 30 and less than 150 characters to get rid of extra noise in the data. A Naive Bayes classifier is trained on this model with a count feature of bag of words (ngram = 1 and 2). Additionally we also use question mark as a high weighted feature. This test is performed on data exacted from the same corpus, not present in the training data. In Table 1, answer refers to any sentence not being a question. We are more concerned about our accuracy of detecting the right question than an answer.

	Precision	**Recall**	**F1**
Answer	0.95	0.95	0.95
Question	0.98	0.98	0.98
Average	0.97	0.97	0.97

Table 1: Question and Answer

Classifier B uses the same amazon data set but we only use training examples that are questions. The two classes that we try to distinguish between are Yes/No question and open-ended question. As in the previous case, we use only sentences more than 30 and less than 150 characters. Further we remove all punctuation marks including question mark. The results are presented in Table 2.

Finally in Classifier C, we further classify open-ended question into Date, Location, Number or Free Text. For this purpose we use data set by University of Illinois [1]. This data has been used in the domain of question answering

	Precision	**Recall**	**F1**
Open Ended	0.80	0.84	0.82
Yes/No	0.86	0.83	0.85
Average	0.83	0.83	0.83

Table 2: Open-Ended and Yes/No question

to perform type identification. For example, "What is the temperature outside?" expects a number as an output. The results are presented in Table 3.

	Precision	**Recall**	**F1**
Date	1.00	0.74	0.85
Free Text	0.84	1.00	0.91
Location	0.98	0.75	0.85
Number	1.00	0.59	0.74
Average	0.83	0.83	0.83

Table 3: Date, Free Text, Location and Number

3.2 UI Mapping

The mapping between classes as predicted by the three classifiers and UI elements is presented in Table 4. After the classification step, we extract the original sentence from the email text with the predicted classes and return it in the form of a list. This service is wrapped in a rest endpoint which accepts a POST request and returns a JSON equivalent of the result.

Class	**UI element**
Yes/No	Radio Buttons
Date	Input Element type = date
Number	Input Element type = number
Location	Map
Free Text	Input Element type = text

Table 4: Class and UI elements

3.3 Chrome Extension

In this section we will look at the chrome extension built specifically for Gmail web client. Chrome extensions allow developers to inject javascript into a web page's environment. There are two different javascript environment working on the same DOM structure. The extension's background page can register for events and create new HTML elements.

We introduce a special button which helps to formulate the reply. This button when clicked calls the server hosting the classifier with a POST message. The message carries the email content. The reply which is in the form of [Text, UI element, Filler] is parsed. A pop-up UI is presented to user

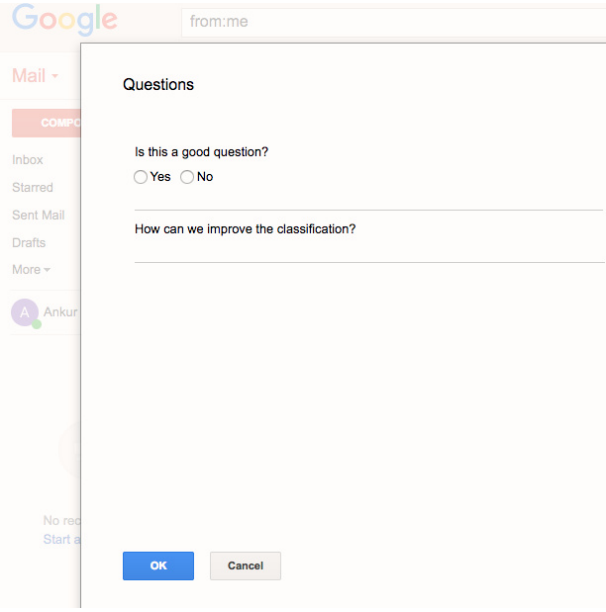

Figure 1: A pop form generated from email.

with key answers to be filled by the user.

4. Future Work

We can improve the current system in many dimensions. For cleansing email, additional pre-processing steps can be employed. For example, removing non ASCII characters and URLs, replace sms-like language with proper English words etc. We can improve the classification by using more features like POS, specific-phrases and matching expected types. Also other machine learning models like Conditional Random Fields can be used. On the UI front, attachments can be added as a new UI element. We can also look at forming partial answers. For example, a question like "How can we improve the classification?" can have a partial response as "The classification can be improved by ...". Finally, an email data set like Enron email data set can be used to test the entire functionality in a corporate environment which generally has email has primary medium of written communication.

5. References

1. Xin Li , Dan Roth, Learning question classifiers: the role of semantic information, Natural Language Engineering, v.12 n.3, p.229-249, September 2006.
2. Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.
3. Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. OReilly Media Inc.
4. Mengting Wan, Julian McAuley, Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems International Conference on Data Mining (ICDM), 2016.
5. Lokesh Shrestha, Kathleen McKeown, Detection of Question-Answer Pairs in Email Conversations, COLING '04 Proceedings of the 20th international conference on Computational Linguistics Article No. 889
6. Baichuan Li et al., Question Identification on Twitter, CIKM '11 Proceedings of the 20th ACM international conference on Information and knowledge management Pages 2477-2480
7. Kyle Dent, Sharoda Paul, Through the twitter glass: detecting questions in micro-text, AAAIWS'11-05 Proceedings of the 5th AAAI Conference on Analyzing Microtext Pages 8-13
8. Helen Kwong, Neil Yorke-Smith, Detection of Imperative and Declarative Question-Answer Pairs in Email Conversations, Journal AI Communications archive Volume 25 Issue 4, October 2012 Pages 271-283
9. Mark Dredze, Tova Brooks, Josh Carroll, Joshua Magarick, John Blitzer, Fernando Pereira. Intelligent Email : Reply and Attachment Prediction.